CKA exam testing

There will be 32 questions, 4 hours, 8 clusters, 10 topics.

Tip: Create an alias for all kubelet commands e.g:
alias kg='kubectl get'
alias kc='kubectl create -f'

Preparation

Q: Create a Job that run 60 time with 2 jobs running in parallel
https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/

Q: Find which Pod is taking max CPU
Use `kubectl top` to find CPU usage per pod

Q: List all PersistentVolumes sorted by their name
Use `kubectl get pv --sort-by=` <- this problem is buggy & also by default kubectl give the output sorted by name.

Q: Create a NetworkPolicy to allow connect to port 8080 by busybox pod only
https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/
Make sure to use `apiVersion: extensions/v1beta1` which works on both 1.6 and 1.7

Q: fixing broken nodes, see
https://kubernetes.io/docs/concepts/architecture/nodes/

Q: etcd backup, see
https://kubernetes.io/docs/getting-started-guides/ubuntu/backups/
https://www.mirantis.com/blog/everything-you-ever-wanted-to-know-about-using-etcd-with-kubernetes-v1-6-but-were-afraid-to-ask/

Q: TLS bootstrapping, see
https://coreos.com/kubernetes/docs/latest/openssl.html
https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/
https://github.com/cloudflare/cfssl

Q: You have a Container with a volume mount. Add a init container that creates an empty file in the volume. (only trick is to mount the volume to init-container as well)

https://kubernetes.io/docs/concepts/workloads/pods/init-containers/
```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
```

```
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  initContainers:
  - name: init-touch-file
    image: busybox
    volumeMounts:
    - mountPath: /data
      name: cache-volume
    command: ['sh', '-c', 'echo "" > /data/harshal.txt']
  volumes:
  - name: cache-volume
    emptyDir: {}
````

Q: When running a redis key-value store in your pre-production environments many deployments are incoming from CI and leaving behind a lot of stale cache data in redis which is causing test failures. The CI admin has requested that each time a redis key-value-store is deployed in staging that it not persist its data.

Create a pod named non-persistent-redis that specifies a named-volume with name app-cache, and mount path /data/redis. It should launch in the staging namespace and the volume MUST NOT be persistent.
Create a Pod with EmptyDir and in the YAML file add namespace: CI

Q: Setting up K8s master components with a binaries/from tar balls:

Also, convert CRT to PEM: openssl x509 -in abc.crt -out abc.pem
- https://coreos.com/kubernetes/docs/latest/openssl.html
-
https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/04-certificate-authority.md
-
https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/08-bootstrapping-kubernetes-controllers.md
- https://gist.github.com/mhausenblas/0e09c448517669ef5ece157fd4a5dc4b
- https://kubernetes.io/docs/getting-started-guides/scratch/
- http://alexander.holbreich.org/kubernetes-on-ubuntu/ maybe dashboard?
- https://kubernetes.io/docs/getting-started-guides/binary_release/
- http://kamalmarhubi.com/blog/2015/09/06/kubernetes-from-the-ground-up-the-api-server/

Q: Find the error message with the string “Some-error message here”.
https://kubernetes.io/docs/concepts/cluster-administration/logging/ see kubectl logs and /var/log for system services

Q 17: Create an Ingress resource, Ingress controller and a Service that resolves to cs.rocks.ch.

First, create controller and default backend
```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress/master/controllers/nginx/examples/default-backend.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress/master/examples/deployment/nginx/nginx-ingress-controller.yaml
```

Second, create service and expose
```
kubectl run ingress-pod --image=nginx --port 80
kubectl expose deployment ingress-pod --port=80 --target-port=80 --type=NodePort
```

Create the ingress
```
cat <<EOF >ingress-cka.yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-service
spec:
  rules:
  - host: "cs.rocks.ch"
    http:
      paths:
      - backend:
          serviceName: ingress-pod
          servicePort: 80
EOF
```

To test, run a curl pod
```
kubectl run -i --tty client --image=tutum/curl
curl -I -L --resolve cs.rocks.ch:80:10.240.0.5 http://cs.rocks.ch/
```

Q: Run a Jenkins Pod on a specified node only.
https://kubernetes.io/docs/tasks/administer-cluster/static-pod/
Create the Pod manifest at the specified location and then edit the systemd service file for kubelet(/etc/systemd/system/kubelet.service) to include
`--pod-manifest-path=/specified/path`. Once done restart the service.

Q: Use the utility nslookup to look up the DNS records of the service and pod.
From this guide, https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/
Look for “Quick Diagnosis”

```
$ kubectl exec -ti busybox -- nslookup mysvc.myns.svc.cluster.local
```

Naming conventions for services and pods:
For a regular service, this resolves to the port number and the CNAME: `my-svc.my-namespace.svc.cluster.local`.

For a headless service, this resolves to multiple answers, one for each pod that is backing the service, and contains the port number and a CNAME of the pod of the form `auto-generated-name.my-svc.my-namespace.svc.cluster.local`
When enabled, pods are assigned a DNS A record in the form of

`pod-ip-address.my-namespace.pod.cluster.local`.

For example, a pod with IP `1.2.3.4` in the namespace `default` with a DNS name of `cluster.local` would have an entry: `1-2-3-4.default.pod.cluster.local`

Q: Start a pod automatically by keeping manifest in /etc/kubernetes/manifests
Refer to https://kubernetes.io/docs/tasks/administer-cluster/static-pod/
Edit kubelet.service on any worker node to contain this flag **`--pod-manifest-path=/etc/kubernetes/manifests`** then place the pod manifest at **`/etc/kubernetes/manifests.`**
Now restart kubelet.

Some other Questions:

1. Main container looks for a file and crashes if it doesnt find the file. Write an init container to create the file and make it available for the main container

2. Install and Configure kubelet on a node to run pod on that node without contacting the api server
3. Take backup of etcd cluster
4. rotate TLS certificates
5.rolebinding
6.Troubleshooting - involved identifying failing nodes, pods , services and identifying cpu utilization of pods.